Emptiness check for Büchi Automata based on decomposition

Etienne Renault

LRDE/LIP6

July 5, 2012
What is Model Checking?

Check if a given system respects the specified behaviour.

We need:
- a system: a microwave oven,
- a property:
  - The oven doesn’t heat up until the door is closed.
  - If start button is pressed, the oven will heat up in the future.

Objectives

Detect if the specified behaviours are correct otherwise return a counterexample leading to the violation of the property.
What is Model Checking?

Check if a given system respects the specified behaviour.

We need:

- a system: a microwave oven,
- a property:
  - The oven doesn’t heat up until the door is closed.
  - If start button is pressed, the oven will heat up in the future.

Objectives

Detect if the specified behaviours are correct otherwise return a counterexample leading to the violation of the property.
What is Model Checking?

Check if a given system respects the specified behaviour.

We need:

- a system: a microwave oven,
- a property:
  - The oven doesn’t heat up until the door is closed.
  - If start button is pressed, the oven will heat up in the future.

Objectives

Detect if the specified behaviours are correct otherwise return a counterexample leading to the violation of the property.
What is Model Checking?

The oven doesn’t heat up until the door is closed.

Counterexample

Emptiness check for BA based on decomposition
Automata-Theoretic LTL Model Checking

High-level model $M$

State-space automaton $A_M$

Product Automaton $A_M \otimes A_{\neg \varphi}$

Negated property automaton $A_{\neg \varphi}$

Synchronized product $L(A_M \otimes A_{\neg \varphi}) = L(A_M) \cap L(A_{\neg \varphi})$

Emptiness check $L(A_M \otimes A_{\neg \varphi}) \neq \emptyset$

$LTL$ property $\varphi$

$LTL$ translation

$M \models \varphi$ or counterexample

Emptiness check for BA based on decomposition
Automata-Theoretic LTL Model Checking

High-level model $M$

State-space automaton $A_M$

Synchronized product $L(A_M \otimes A_{\neg \varphi}) = L(A_M) \cap L(A_{\neg \varphi})$

Product Automaton $A_M \otimes A_{\neg \varphi}$

Emptiness check

Negated property automaton $A_{\neg \varphi}$

$L(A_M \otimes A_{\neg \varphi}) \neq \emptyset$

Emptiness check for BA based on decomposition

$LTL$ property $\varphi$

$LTL$ translation

$M \models \varphi$ or counterexample
Automata-Theoretic LTL Model Checking

High-level model $M$

State-space automaton $A_M$

Negated property automaton $A_{\neg \varphi}$

Synchronized product $L(A_M \otimes A_{\neg \varphi}) = L(A_M) \cap L(A_{\neg \varphi})$

Product Automaton $A_M \otimes A_{\neg \varphi}$

Emptiness check $L(A_M \otimes A_{\neg \varphi}) \neq \emptyset$

$M \models \varphi$ or counterexample

Emptiness check for BA based on decomposition
Automata-Theoretic LTL Model Checking

High-level model $M$

State-space automaton $A_M$

Product Automaton $A_M \otimes A_{\neg \varphi}$

Synchronized product $\mathcal{L}(A_M \otimes A_{\neg \varphi}) = \mathcal{L}(A_M) \cap \mathcal{L}(A_{\neg \varphi})$

Negated property automaton $A_{\neg \varphi}$

Emptiness check $\mathcal{L}(A_M \otimes A_{\neg \varphi}) \neq \emptyset$

$LTL$ property $\varphi$

$LTL$ translation

$M \models \varphi$ or counterexample

Emptiness check for BA based on decomposition
Automata-Theoretic LTL Model Checking

High-level model $M$

State-space automaton $A_M$

Product Automaton $A_M \otimes A_{\neg \varphi}$

Synchronized product
$L(A_M \otimes A_{\neg \varphi}) = L(A_M) \cap L(A_{\neg \varphi})$

Negated property automaton $A_{\neg \varphi}$

Emptiness check
$L(A_M \otimes A_{\neg \varphi}) \neq \emptyset$

$LTL$ property $\varphi$

$LTL$ translation

Emptiness check for BA based on decomposition

$M \models \varphi$ or counterexample
Accepting runs are infinite sequences visiting infinitely often each acceptance conditions.
Categories of Automata

Terminal Automaton

Weak Automaton

Strong Automaton

Reachability

DFS

NDFS

Emptiness check for BA based on decomposition
Properties of the synchronized product

- High-level model $M$
- State-space automaton $A_M$
- Synchronized product
  \[ L(A_M \otimes A_{\neg \varphi}) = L(A_M) \cap L(A_{\neg \varphi}) \]
- Product Automaton $A_M \otimes A_{\neg \varphi}$
- Emptiness check
  \[ L(A_M \otimes A_{\neg \varphi}) \neq \emptyset \]
- LTL property translation
- LTL property $\varphi$
- Negated property automaton $A_{\neg \varphi}$
- Emptiness check for BA based on decomposition

$M \models \varphi$ or counterexample
Global approach

- Approach proposed by Somenzi, Bloem and Ravi (CAV’99).
- Over-approximate class syntactically (from the formula).
- Apply the more efficient emptiness check algorithm.

Starting point

Automata can be composed of subautomata of each type, how can we use this information to perform efficient emptiness check?
Our works

- Decide class structurally (from the automaton).
- Decompose this automaton.
- Each emptiness checks can be launched in parallel.
Our works

- Decide class structurally (from the automaton).
- Decompose this automaton.
- Each emptiness checks can be launched in parallel.
Our works

- Decide class structurally (from the automaton).
- Decompose this automaton.
- Each emptiness checks can be launched in parallel.
Example of Decomposition for \((G a \rightarrow G b) \land c\)
As soon as a counterexample is found, kill other emptiness checks.

Otherwise, wait for the end of all emptiness checks.
Models: Ring, Fms, Kanban, Philo
- 2600 formulas
- 427 empty result
- 2173 counterexamples found

Emptiness check for BA based on decomposition
Conclusion and future works

- Minimising the original automaton by composing all minimized subautomata.
- Extracting other automata.
- Mixing decomposition with symbolic, explicit and hybrid approaches.
- Considering other temporal logics (PSL is already supported).
- Mixing this approach with other type of automata (Streett, testing automata,...).
- Mixing this with other techniques of verification (Partial Order, SAT,...).
That’s all folks...

Questions?